REMARKS

Claims 1-6, 8-13, 15, and 17-18 are pending. Applicant respectfully traverses and requests reconsideration. Support for new claims 19 and 20 may be found in the specification at least on page 4, lines 1-10 and on page 5, lines 15-31. As such, no new matter is added. The undersigned would like to thank the Examiner for the inteview on May 6, 2004, and for an indication that the language in at least new claims 19 and 20 appears to overcome the current rejections.

I.   Summary of the Examiner's Objections/Rejections

Claims 1-6, 8-13, 15, and 17-18 remain rejected under 35 U.S.C. §102(b) based on Favor ("Favor") (WO 97/13194).

FAVOR

Favor is directed to converting Common x86 instructions by instruction decode hardware to operations in an internal RISC86 instruction set. (Favor, p. 3 lines 36-37, emphasis added). Favor describes an internal RISC-type instruction format that facilitates the translation of a very large number of CISC-type instructions into a small number of RISC-type operations. (Favor, p. 1, lines 38-39). One problem, described by Favor, with the usage of lookup ROM for decoding x86 instructions is that the process of accessing a microprogram control store is inherently slower and less efficient than hardwired translation of instructions. (Favor, p. 1, lines 28-30). Favor is therefore directed to hardware logic using the internal RISC instruction format that permits more common CISC-type instructions to be converted, as compared to conversion via lookup ROM. (Favor, p. 1, lines 41-42). The instruction set includes a plurality of instruction codes arranged in a fixed bit length structure. (Favor, p. 2, lines 11-12). A regular fixed bit length structure greatly reduces circuit complexity and size and substantially increases efficiency. (Favor, p. 2, lines 28-30).

Instructions from main memory 130 are loaded into instruction cache 214 via a predecoder 270 for anticipated execution. (Favor, p. 4, lines 9-11). The predecoder 270 generates predecode bits that are stored in combination with instruction bits in the instruction cache 214. (Favor, p. 4, lines 11-12).

X86 instructions typically include an opcode followed by a modr/m byte. (Favor, p. 34, lines 28). The modr/m byte designates an indexing type or register number to be used in the instruction. (Favor, p. 34, lines 28-29). Figure 6A is a register operation (RegOp) field 610 encoding graphic that illustrates various fields in the RegOp format. (Favor, p. 35, lines 1-2). The RegOp field 610 also includes a single-bit set status (SS) field 624 at bit location [9]. (Favor, p. 35, lines 7-8). For Ops in which the set status (SS) field 624 is set to 1, indicating that this Op does modify flags, the extension field (EXT) 614 specifies four status modification bits designating the groups of flags that are modified by the Op. (Favor, p. 36, lines 25-27).

Decoupling of condition code handling from operation type, using the independent TYPE 612 and set status (SS) field 624, allows some operations to be defined that do not update the flags. (Favor, p. 38, lines 7-9). Accordingly, for those circumstances in which updating of condition flags *is not necessary*, it is highly advantageous to disable flag updating to avoid unnecessary dependency on previous flag values. (Favor, p. 38, lines 9-11). Therefore, Favor explicity teaches disabling flag updating in which updating of condition flags is not necessary to avoid unnecessary dependency on previous flag valves. The EXT field 614 is used to update condition flags, including six flags corresponding to x86 flags and two emulation flags. (Favor, p. 38, lines 1-2). (emphasis added)

35 U.S.C. §102(b) Rejections

Claims 1-6, 8-13, 15 and 17-18 are rejected under 35 U.S.C. §102(b) over Favor (WO 97/13194). A claim is anticipated only if each and every element arranged as required by the claim is found in a single prior art reference.[1]

Independent Claims 1 and 10

As to Claims 1 and 10, these claims require, among other things, "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code." As such, claims 1 and 10 recite "at least one flag modification enabled bit allows updating of at least one flag" rather than disabling flag updating in which updating of condition flags is not necessary to avoid unnecessary dependency on previous flag valves.

---

[1] *Verdgaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 U.S.P.Q. 2d 1051, 1053 (Fed. Cir. 1987); *In re Bond*, 910 F.2d 831, 15 U.S.P.Q. 2d 1566 (Fed. Cir. 1990); M.P.E.P. 2131.

The Office Action continues to assert that "inherently, Favor's system must determine whether bit 624 is set when processing 612-type instructions and then will update at least one flag when the bit is set." However, Favor on page 38, lines 7-11 states:

> Decoupling of condition code handling from operation type, using the independent type 612 and set status (SS) field 624, allows some operations to be defined which do not update the flags. Accordingly, <u>for those circumstances in which updating of condition flags is not necessary,</u> it is highly advantageous to disable flag updating to avoid unnecessary dependency on previous flag values.

(Emphasis added) As a result, Favor explicitly teaches disabling flag updating under only one condition, namely "for those circumstances in which updating of condition flags is not necessary." Favor explicitly teaches that the reason for disabling flag updating is only "for those circumstances in which updating of condition flags is not necessary" because Favor teaches avoiding unnecessary updating of condition flags. Since Favor explicitly teaches disabling flag updating only to avoid unnecessary dependency on previous flag values, then Favor explicitly teaches away from disabling flag updating when non-native instructions share a common non-native instruction flag. Therefore, Favor is limited to disabling flag updating only when non-native instructions do not share a non-native instruction flag. As a result, Favor fails to teach "emulating non-native instructions using native instructions containing the flag modification enable bit." Since Favor explicitly teaches away from emulating non-native instructions that share a common flag modification enable bit, Favor explicitly teaches away from "emulating non-native instructions using native instructions containing the flag modification enable bit." Further, Favor teaches away from "receiving at least two non-native instructions containing data representing operation code and at least one non-native instruction flag" as taught in claim new 19.

The Office Action fails to show where Favor teaches "updating at least one flag in response to determining a status of the at least one flag modification enabled bit" because Favor fails to teach "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operation code." In other words, not only does the Office Action fail to show where Favor teaches "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operation code," Favor also necessarily fails to teach "updating at least one flag in response to

away from "updating at least one flag in response to determining a status of the at least one flag modification enabled bit" because the flag modification enabled bit as taught by Favor is related to the fixed bit-length structure rather than the non-native instruction." For example, as admitted in the Office Action on page 3, paragraph 6, FIG. 6a shows the RISC-type instruction format, including the single bit set status (SS) field 624 at bit location [9], as cited in Favor on page 35, lines 7-8. The Applicant would like to emphasize that Favor is teaching the presence of the single bit set status field in the RISC-type instruction (fixed bit-length) rather than in the non-native instruction. As a result, the Office Action fails to show where Favor teaches each and every element of the claims, including "receiving at least one instruction containing data representing operational code and data representing at least one flag modification enabled bit emulating non-native instructions using native instruction containing the flag modification enabled bit." As such, the Office Action continues to ignore explicit elements in the claims.

The Favor language, as cited at p. 36, lines 25-26, which states "[f]or Ops in which the set status (SS) field 624 is set to 1, indicating that this Op does modify flags, the extension field (EXT) 614 specifies four status modification bits designating the groups of flags that are modified by the Op," is limited to merely indicating that this Op does modify flags for Ops in which the set status (SS) field 624 is set to 1, rather than "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code." Applicant would like to point out the distinction between merely indicating "that this Op does modify flags for Ops in which the set status (SS) field 624 is set to 1" and "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code." Applicant would also like to point out the distinction between the conditions "determine whether bit 624 is set when processing 612-type instructions" and "in response to executing the operational code."

Further, Favor is directed to solving an entirely different problem, namely providing a regular structure that greatly reduces circuit complexity and size and substantially increases efficiency. (Favor, page 2, lines 28-30). As a result, Favor has no need for "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code." Accordingly, Applicant requests a showing of where Favor teaches "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code."

Rather than teach "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code," Favor teaches "Common x86 instructions are converted by instruction **decode hardware** to operations in an internal RISC86 instruction set." (Favor, page 3, lines 36-37, emphasis added). As a result, Favor teaches that the conversion is performed in decode hardware rather than by hardware that would execute the instruction and, therefore, Favor teaches away from "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code." For example, Favor teaches various condition codes (i.e., "WRFLG conditional Op", page 36, line 28; "condition codes," page 36, line 37 – page 37, line 18 "branch conditions," page 37, lines 26-35). Favor explicitly teaches that the *decoder* then changes the branch prediction, as described at page 37, lines 35-36, rather than "in response to executing the operational code." Favor also teaches generating predecode bits in predecoder 270. (Favor, p. 4, lines 9-11). Consequently, rather than describe "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code", Favor instead teaches indicating that this Op does modify flags as a result of decode, as opposed to "in response to executing the operational code." As a result, Favor teaches away from the claims.[2]

The Favor language, as cited at p. 35, lines 7-8, which states "[t]he RegOp field 610 also includes a single-bit set status (SS) field 624 at bit location [9]", is limited to a single-bit set status (SS) field 624 located within the RegOp field 610, rather than "receiving at least one instruction containing data representing operational code and data representing at least one flag modification enable bit." Applicant would like to point out, among other things, the distinction between "an instruction containing data representing operational code and data representing at least one flag modification enable bit" and "[t]he RegOp field 610 also includes a single-bit set status (SS) field 624." Since Favor requires that the RegOp field include the SS field, Favor does not teach that the instruction contains data representing operational code and data representing at least one flag modification enable bit, as arranged in the claims.

---

[2] A prior art reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention. *W.L. Gore & Associates, Inc. v. Garlock, Inc.*, 721 F.2d 1540, 220 U.S.P.Q. 303 (Fed. Cir. 1983), *cert. denied,* 469 U.S. 851 (1984), M.P.E.P. 2141.02.

10

The Applicant cannot find where the cited portion of Favor teaches each and every element as arranged in the claims, including "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code." Accordingly, the Applicant requests a showing of "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code." Consequently, as the Office Action has similarly ignored a principal limitation of Claims 1 and 10, namely "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code" and since the Office Action does not disclose how Favor teaches "determining whether the at least one flag modification enabled bit allows updating of at least one flag in response to executing the operational code," as defined in Claims 1 and 10, Applicant submits that Favor does not anticipate the invention as defined in Claims 1 and 10. As a result, Favor fails to teach each and every element as arranged in the claims and, therefore, the rejection is improper.

Dependent Claims 4 and 13

Applicant further submits that Claims 4 and 13 are also allowable in light of the presence of novel and non-obvious elements contained in Claims 4 and 13 that are not otherwise present in Claims 1 and 10, respectively. Applicant also submits that Claims 4 and 13 depend from Claims 1 and 10, respectively, and as dependent therefrom, Claims 4 and 13 are allowable for at least the reasons Claims 1 and 10, respectively, are allowable. Applicant requests a showing of where Favor teaches "updating a flag register if the flag modification enable bit is set to allow modification of a flag in the flag register," as arranged in the claims.

Dependent Claim 5

Favor, as cited, teaches indicating that this Op does modify flags for Ops in which the set status (SS) field 624 is set to 1, rather than "evaluating the flag modification enable bit to preserve flag bit settings for variable length instructions that are emulated using the fixed native instructions." Applicant would like to point out the distinction between merely indicating "that this Op does modify flags for Ops in which the set status (SS) field 624 is set to 1" and "evaluating the flag modification enable bit to preserve flag bit settings for variable length instructions that are emulated using the fixed native instructions." Applicant requests a showing

of where Favor teaches "evaluating the flag modification enable bit to preserve flag bit settings for variable length instructions that are emulated using the fixed native instructions."

Applicant further submits that Claim 5 is also allowable in light of the presence of novel and non-obvious elements contained in Claim 5 that are not otherwise present in Claim 1. Applicant also submits that Claim 5 depends from Claim 1, and as dependent therefrom, Claim 5 is allowable for at least the reasons Claim 1 is allowable.

Dependent Claims 9 and 18

With respect to the rejections of Claims 9 and 18, the previous Office Action generally cites to Favor from page 35, line 1 through page 38, line 11 without specifically identifying where Favor teaches each and every element as arranged in the claims. As such, the Applicant respectfully requests that the Office Action show the particular part of Favor relied on for rejecting each and every claim in the application including Claims 9 and 18 in compliance with 37 C.F.R. § 1.104(c)(2). For example, Claims 9 and 18 essentially recite "emulating unconverted variable length X86 instructions using a plurality of native instructions wherein the native instructions include the at least one flag modification enabled bit set to prevent changing of non-native instruction flags in response to execution of the plurality of native instructions." Specifically, the portion of Favor previously cited at page 38, lines 9-11, which states, "Accordingly, for those circumstances in which updating of condition flags is not necessary, it is highly advantageous to disable flag updating to avoid unnecessary dependency on previous flag values," is limited to disabling flag updating "for those circumstances in which updating of condition flags is not necessary," rather than "wherein the native instructions include the at least one flag modification enabled bit set to prevent changing of non-native instruction flags in response to execution of the plurality of native instructions." The Applicant would like to point out the distinction between disabling flag updating "for those circumstances in which updating of condition flags is not necessary" and "wherein the native instructions include the at least one flag modification enabled bit set to prevent changing of non-native instruction flags in response to execution of the plurality of native instructions." As such, instead of preventing the changing of non-native instruction flags in response to execution of the plurality of native instructions that use the non-native instruction flags, Favor teaches the opposite approach, namely disabling flag updating "for those circumstances in which updating of condition flags is not necessary." (Favor page 38, lines 9-10). As a result, Favor teaches away from the claims because Favor explicitly

teaches disabling flag updating only "for those circumstances in which updating of condition flags is not necessary" rather than as claimed, "wherein the native instructions include the at least one flag modification enabled bit set to prevent changing of non-native instruction flags in response to execution of the plurality of native instructions." Further, the claims explicitly teach to prevent changing of non-native instruction flags, rather than to prevent changing of native instruction flags, as explicitly taught in Favor at page 38, lines 8-11.

As stated above, Favor, understood as cited, is limited to converting X86 instructions by decode hardware to operations in an internal RISC86 instruction set. Applicant cannot find where Favor as cited teaches, among other things, "emulating unconverted variable length X86 instructions." Applicant submits that the Office Action fails to show where Favor teaches where length X86 instructions are unconverted. Accordingly, Applicant requests a showing of where Favor teaches:

> converting variable length X86 instructions to a plurality of native instructions wherein the plurality of native instructions include that at least one flag modification enable bit set to allow changing of non-native instruction flags in response to execution of the plurality of native instructions; and

> emulating unconverted variable length X86 instructions using a plurality of native instructions wherein the native instructions include the at least one flag modification enable bit set to prevent changing of non-native instruction flags in response to execution of the plurality of native instructions.

Applicant further submits that Claims 9 and 18 are also allowable in light of the presence of novel and non-obvious elements contained in Claims 9 and 18 that are not otherwise present in Claims 1 and 10, respectively. Applicant also submits that Claims 9 and 18 depend from Claims 1 and 10, respectively, and as dependent therefrom, Claims 9 and 18 are allowable for at least the reasons Claims 1 and 10, respectively, are allowable.

Dependent Claims 3, 6, 8, 11, 12, 15, and 17

Applicant further submits that Claims 3, 6, 7, 8, 11, 12, 15, and 17 are also allowable in light of the presence of novel and non-obvious elements contained in these claims that are not otherwise present Claims 1 and 10. Applicant also submits that these Claims depend from

Claims 1 and 10, and as dependent therefrom, Claims 3, 6, 7, 8, 11, 12, 15, and 17 are allowable for at least the reasons Claims 1 and 10 are allowable.

New Claims 19 and 20

Applicant asserts the relevant arguments above. As previously stated, Favor explicitly teaches disabling flag updating under only one condition, namely "for those circumstances in which updating of condition flags is not necessary." Favor explicitly teaches that the reason for disabling flag updating is only "for those circumstances in which updating of condition flags is not necessary" because Favor teaches avoiding unnecessary updating of condition flags. As previously stated, Favor does not teach and further teaches away from "receiving at least two non-native instructions containing data representing operation code and at least one non-native instruction flag." As such, the Office Action fails to show how Favor describes each and every element as arranged in Claim 19. Further, Claims 19 and 20 are allowable in light of the presence of novel and nonobvious elements contained in these claims that are not otherwise present in Claim 1 and 10.

Applicant also submits that Claim 20 depends from Claim 19 and as a dependant claim therefrom, Claim 20 is allowable for at least the reasons Claim 19 is allowable. Further, Claim 19 recites "preventing updating of at least one flag in response to determining whether the at least one flag modification enable bit allows updating of at least one flag." As previously stated, Favor on page 38, lines 7-8 explicitly teaches disabling flag updating only "for those circumstances in which updating of condition flags is not necessary." As such, since Favor explicitly teaches disabling flag updating only for those circumstances in which updating of condition flags is not necessary, Favor does not teach and further teaches away from

"determining whether the at least one flag modification enable bit allows updating of at least one flag." For at least these reasons, Claims 19 and 20 are allowable.

Applicant respectfully requests that the pending claims be allowed to issue. The Examiner is invited to contact the below-listed attorney if the Examiner believes that a telephone conference will advance the prosecution of this Application.

Respectfully submitted,

Date: June 18, 2004

By: _____
Themi Anagnos, Reg. No. 47,388

Vedder, Price, Kaufman & Kammholz, P.C.
222 North LaSalle Street, Suite 2600
Chicago, Illinois 60601
PHONE: (312) 609-7970
FAX: (312) 609-5005